

## REMARKS

This is intended as a full and complete response to the Office Action dated August 20, 2008, having a shortened statutory period for response set to expire on November 20, 2008. Please reconsider the claims pending in the application for reasons discussed below.

Claims 1-13 are pending in the application. Claims 1-13 remain pending following entry of this response. Claims 1, 6, and 8 have been amended. Applicants submit that the amendments do not introduce new matter.

Further, Applicants are not conceding in this application that those amended claims are not patentable over the art cited by the Examiner, as the present claim amendments are only for facilitating expeditious prosecution of the claimed subject matter. Applicants respectfully reserve the right to pursue these (pre-amended claims) and other claims in one or more continuations and/or divisional patent applications.

### Interview Summary

On November 12, 2008, a telephonic interview was held between Gero McClellan (an attorney of record for Applicants), Johnny Lam (attorney for Applicants), and Examiner Ben Wang.

Applicants discussed the applicability of *Spertus et al.* (U.S. Pat. No. 6,938,245; hereinafter *Spertus*) to claims 1, 6, and 8. Further, Applicants discussed proposed amendments to claims 1, 6, and 8. Although no agreement could be reached at the time of the interview, the Examiner indicated that the proposed amendments would likely overcome the rejections of record, pending further review and consideration. The proposed amendments are reflected herein.

### Claim Rejections - 35 U.S.C. § 102

Claims 1-8 and 11-13 are rejected under 35 U.S.C. 102(e) as being anticipated by *Spertus*.

Applicants respectfully traverse this rejection.

"A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987). "The identical invention must be shown in as complete detail as is contained in the ... claim." *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 1236, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989). The elements must be arranged as required by the claim. *In re Bond*, 910 F.2d 831, 15 USPQ2d 1566 (Fed. Cir. 1990).

In this case, *Spertus* does not disclose "each and every element as set forth in the claim." For example, *Spertus* does not disclose "allowing a user to establish, via the computer user interface, a relationship between one or more of the memory deallocators and one or more of the memory allocators, wherein the relationship requires that memory space allocated by the one or more allocators is freed by the one or more deallocators." The Examiner suggests that this limitation is disclosed by *Spertus*, Fig. 9; col. 15, lines 20-42. Specifically, the Examiner asserts as follows:

As to claim 1, *Spertus* discloses . . . allowing a user to establish, via the computer interface, a relationship between one or more of the memory deallocators and one or more of the memory allocators, wherein the relationship requires that memory space allocated by the one or more allocators is freed by the one or more deallocators (e.g., Fig. 9; Col. 15, Lines 20-42 – Fig. 9 shows the structure of a bucket 901. Each bucket 901 is made up of a page information structure 907 which includes information 904 indicating the size of the objects that will be allocated from the bucket and a free list pointer 906 indicating the head of the list of unallocated object 911 contained in bucket 901 ...) . . .

Office Action, page 3 (emphasis original). However, the cited portions of *Spertus*, and in fact *Spertus* as a whole, fail to disclose anything at all about "a relationship between one or more of the memory deallocators and one or more of the memory allocators." The cited portions, as well as Figure 9, of *Spertus* are reproduced below:

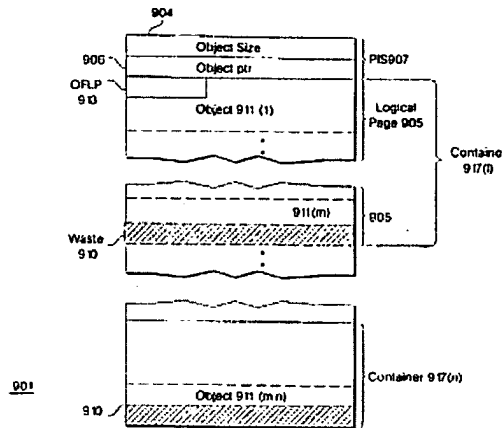


Fig. 9

FIG. 9 shows the structure of a bucket 901. Each bucket 901 is made up of a page information structure 907 which includes information 904 indicating the size of the objects that will be allocated from the bucket and a free list pointer 906 indicating the head of the list of unallocated objects 911 contained in bucket 901. The objects 911 are stored in one or more containers 917, which are made up of one or more contiguous logical pages 905. The size of the container is the number of logical pages 905 which reduces waste 910 to a minimum for the size of object 911 stored in the bucket. Each object 911(i) which has not yet been allocated has an object free list pointer 913 pointing to the next unallocated object 911. When an object is allocated, the free list pointer 913 is overwritten by data and free list pointer 906 is set to point to the next unallocated object 911 in the list. When an object is freed, the allocator to which the bucket belongs links the freed object in at the head of the free list. This has the advantage that an object has been used tends to be reused before it is removed from the caches in the machine that the program that uses the object is executing on. When a bucket 901 runs out of unallocated objects 911, the allocator that uses the bucket obtains an additional container 917 for the bucket from the large object allocator.

*Spertus*, Fig 9; col. 15, lines 20-42. In other words, the Examiner cites a portion of *Spertus* that discusses the structure of a bucket used to facilitate heap defragmentation. Such a discussion about heap defragmentation, and in fact *Spertus* as a whole, fails to disclose anything at all about a relationship between a memory deallocator and a memory allocator. In particular, *Spertus* makes no reference of any kind to establishing relationships between deallocators and allocators. In fact, a simple word search of *Spertus* reveals no instances of the words "relationship" or "deallocator". Thus, contrary to the Examiner's suggestion, *Spertus* does not disclose "a relationship between one or

more of the memory deallocators and one or more of the memory allocators.” Accordingly, Applicants submit that the rejection is defective and should be withdrawn.

Further, for the reasons set forth above, it also follows that *Spertus* does not disclose “upon a call to the one or more deallocators to free a memory space, determining whether the relationship is violated.” The Examiner suggests that *Spertus* discloses this limitation in *Spertus*, Fig. 8; col. 2, lines 59-62; and col. 5; lines 51-67. Specifically, the Examiner asserts as follows:

allowing the code to execute; upon a call to the one or more deallocators to free a memory space, determining whether the relationship is violated (e.g., Fig. 8, element 811 - gcLogAllLeaks - ... write all leaks to the log; Col. 2, Lines 59-62 ... provides memory debugging information such as memory allocations, memory leaks, and current heap size; Col. 5, Lines 51-67 - ... One problem detected by a memory debugger is memory 'leaks' ... ); and

Office Action, page 3. In other words, the Examiner cites portions of *Spertus* that discuss memory debugging information, such as memory allocations and memory leaks. Such a discussion about memory allocations and memory leaks fails to disclose anything at all about “determining whether a relationship [between a deallocator and an allocator] is violated.” Even more problematic is the Examiner’s suggestion that a discussion of memory leaks teaches “upon a call to the one or more deallocators.” To illustrate, *Spertus* describes a memory leak as follows:

One problem detected by a memory debugger is memory "leaks", which occur when a program contains code that allocates memory, but does not contain code that frees the allocated memory when it is no longer being used by the program. Leaks of course always waste memory; with serious leaks, all of the heap memory available to the program may be occupied by leaks, and in that case, the program will fail when a new allocation is attempted.

*Spertus*, col. 5, lines 57-65 (emphasis added). Put another way, memory leaks only occur when a deallocator is never called, even when memory is no longer being used. Therefore, the Examiner’s suggestion that “when a deallocator is never called” teaches “upon a call to the one or more deallocators” is wholly contradictory and utterly untenable. Accordingly, Applicants submit that the rejection is defective and should be withdrawn.

During the telephonic interview, the Examiner indicated that the proposed amendments would likely overcome the rejections of record, pending further review and consideration. Applicants are not conceding in this application that those amended claims are not patentable over the art cited by the Examiner. Nevertheless, in the interest of facilitating prosecution, Applicants have amended claims 1, 6, and 8 to reflect the proposed amendments. Applicants submit that claims 1, 6, and 8, as amended, are not taught by *Spertus*. Accordingly, Applicants respectfully request that the rejection be withdrawn.

Therefore, the claims are believed to be allowable, and allowance of the claims is respectfully requested.

Claim Rejections - 35 U.S.C. § 103

Claims 9-10 are rejected under 35 U.S.C. 103(a) as being unpatentable over *Spertus* in view of *Kolawa et al.* U.S. Pat. No. 5,842,019 (hereinafter *Kolawa*).

Applicants respectfully traverse this rejection.

Claims 9-10 depend from Claim 8. For the reasons given above with respect to Claim 8, Applicants submit that Claims 9-10 are also necessarily allowable.

Therefore, the claims are believed to be allowable, and allowance of the claims is respectfully requested.

Conclusion

Having addressed all issues set out in the office action, Applicants respectfully submit that the claims are in condition for allowance and respectfully request that the claims be allowed.

Respectfully submitted, and  
**S-signed pursuant to 37 CFR 1.4,**

/Gero G. McClellan, Reg. No. 44,227/

---

Gero G. McClellan  
Registration No. 44,227  
PATTERSON & SHERIDAN, L.L.P.  
3040 Post Oak Blvd. Suite 1500  
Houston, TX 77056  
Telephone: (713) 623-4844  
Facsimile: (713) 623-4846  
Attorney for Applicant(s)